УДК 004.04 DOI: 10.18698/2541-8009-2023-6-903

ПРОГРАММНАЯ ПОДСИСТЕМА ТЕСТИРОВАНИЯ ЗНАНИЙ ЯЗЫКОВ ОПИСАНИЯ АППАРАТУРЫ

С.В. Астахов fzastahov@gmail.com
Н.В. Лапшин nikita.lapshin2000@gmail.com

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Статья посвящена разработке программной подсистемы для тестирования знаний языков описания аппаратуры, которая предоставляет возможности по управлению учебными материалами и автоматической проверке заданий, в том числе заданий на описание аппаратных устройств на языке Verilog. Проведен анализ существующих систем тестирования знаний, в ходе анализа сформулированы функциональные требования и составлена диаграмма вариантов использования программной подсистемы для тестирования знаний языков описания аппаратуры. Спроектированы архитектура и компоненты подсистемы. При разработке использована микросервисная архитектура, большинство микросервисов реализовано на языке программирования Golang. Кроме того, для работы с временными диаграммами применяли язык Python и библиотеку PyDigitalWaveTools. Для симуляции поведения цифровых устройств использована программа Icarus Verilog. Проведено функциональное и нагрузочное тестирование разработанной подсистемы.

Ключевые слова

Тестирование знаний, язык описания аппаратуры, HDL, Verilog, система дистанционного обучения, образовательный портал, микросервисная архитектура, Golang, функциональное тестирование, нагрузочное тестирование

Поступила в редакцию 01.06.2023

Введение. В настоящее время существует множество образовательных ресурсов, посвященных тематике информационных технологий. Несмотря на это на данный момент в открытом доступе наблюдается дефицит ресурсов, посвященных изучению языков описания аппаратуры. Среди известных образовательных платформ существует лишь несколько таких, на которых можно настроить автоматическую проверку заданий на написание исходного кода на языке Verilog (или каком-либо другом языке описания аппаратуры) непосредственно в рамках веб-приложения. Однако процесс настройки весьма сложен, поэтому авторы курсов редко используют описанную возможность. В статье рассмотрен процесс разработки программной подсистемы для тестирования знаний языков описания аппаратуры, которая предоставляет возможности по управлению учебными материалами и автоматической проверке заданий (в том числе заданий на описание аппаратных устройств на языке Verilog).

Анализ существующих систем тестирования знаний. По результатам анализа существующих систем тестирования знаний, таких как Huawei University, Coursera, Stepik и Moodle, авторами была предложена классификация методов тестирования знаний (табл. 1), составленная на основе классификации методов тестирования знаний, применяемых в системе Moodle [1].

Таблица 1

Классификация методов тестирования знаний

№ п/п	Тип	Подтип
1	Тестирование с ответом в закрытой форме	1.1. Выбор одного ответа 1.2. Выбор множественных ответов 1.3. Сопоставление
2	Тестирование с коротким ответом	2.1. С автоматизированной проверкой 2.2. С проверкой преподавателем 2.3. С перекрестной проверкой
3	Тестирование с ответом в форме эссе	3.1. С проверкой преподавателем 3.2. С перекрестной проверкой
4	Тестирование на написание исходного кода	4.1. С проверкой по референсным значениям 4.2. Автоматизированное тестирование на проверяющей стороне 4.3. Другие

В последующем из рассмотренных методов тестирования знаний были выделены наиболее подходящие для использования в подсистеме для тестирования знаний языков описания аппаратуры методы. Кроме того, были предложены типы обратной связи, предоставляемой обучающемуся, в случае допущения им ошибки при решении задания (табл. 2).

Таблица 2

Методы тестирования знаний в разработанной подсистеме

№ п/п	Тип	Подтип	Вид обратной связи
1	Тестирование с ответом в закрытой форме	Выбор одного ответа	Текстовое пояснение ошибки
		Выбор нескольких ответов	Информации о наличии ложноположительных (ложноотрицательных) ответов
2	Задание на написание исходного кода	Автоматизированное тестирование на проверяющей стороне	Информация о несоответствующих сигналах

По результатам проведенного анализа была составлена диаграмма вариантов использования подсистемы для тестирования знаний языков описания аппаратуры, представленная на рис. 1 [2].

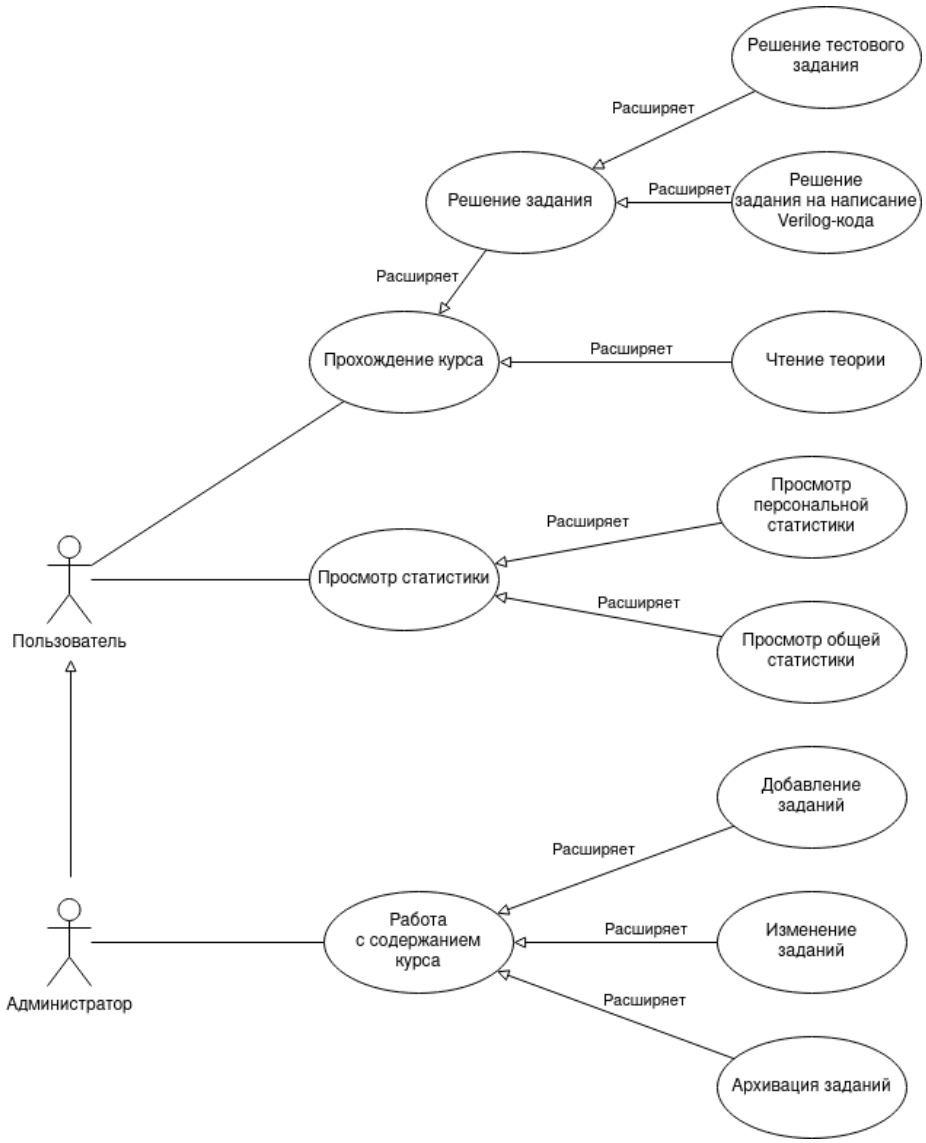

Рис. 1. Диаграмма вариантов использования подсистемы

Проектирование архитектуры подсистемы. После анализа аналогов необходимым этапом является разработка архитектуры. Разработанную подсистему предполагается использовать как информационную системы образовательного портала, что отражено в архитектуре информационной системы, показанной на контекст-диаграмме (нотация C4) на рис. 2 [3].

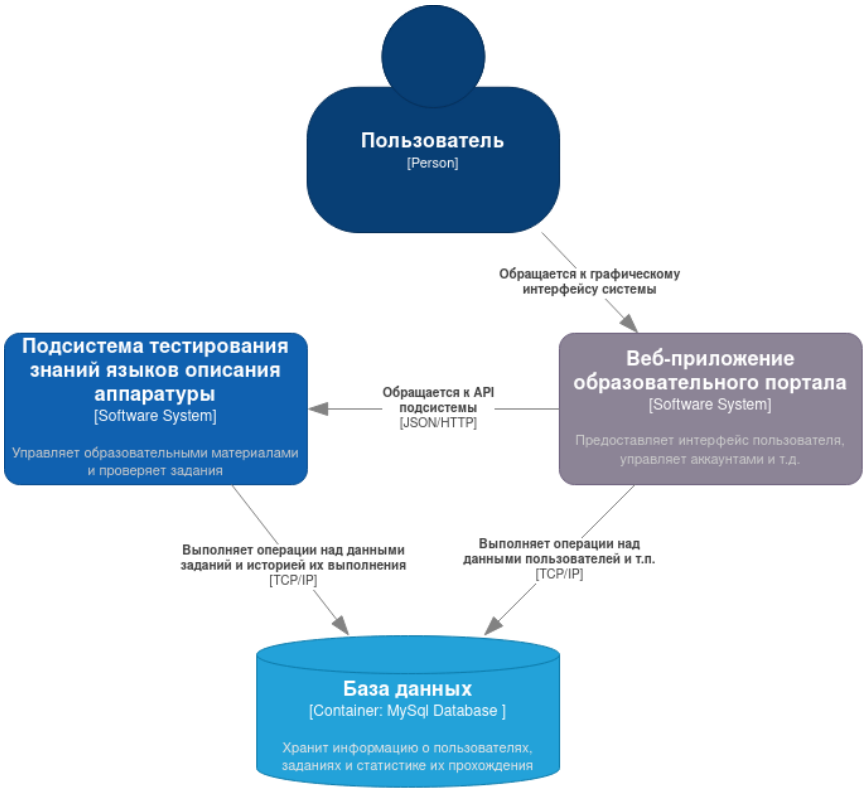

Рис. 2. Контекст-диаграмма информационной системы

Поскольку разработанная подсистема является весьма сложной, выполняемые в ней операции разнородны, может потребоваться использование различных языков и библиотек, а некоторые из операций могут занимать значительное время, при разработке было решено использовать микросервисную архитектуру [4].

На основе функциональных требований и представленной ранее диаграммы вариантов использования была разработана структура компонентов подсистемы, включающая следующие микросервисы:

– микросервис взаимодействия с БД — реализует CRUD-операции над данными в БД;

– микросервис анализа решений (анализатор) — выполняет проверку и анализ пользовательских решений;

– микросервис синтеза устройств (синтезатор) — выполняет синтез устройств из Verilog-кода и симулирует их работу;

– микросервис разбора временных диаграмм, микросервис генерации временных диаграмм wavedrom — преобразуют временные диаграммы в удобные для хранения и обработки форматы;

– микросервис анализа статистики;

– основной микросервис — реализует верхнеуровневую логику подсистемы, связывает остальные микросервисы.

Детализированная архитектура разработанной подсистемы показана на контейнер-диаграмме (нотация C4) на рис. 3.

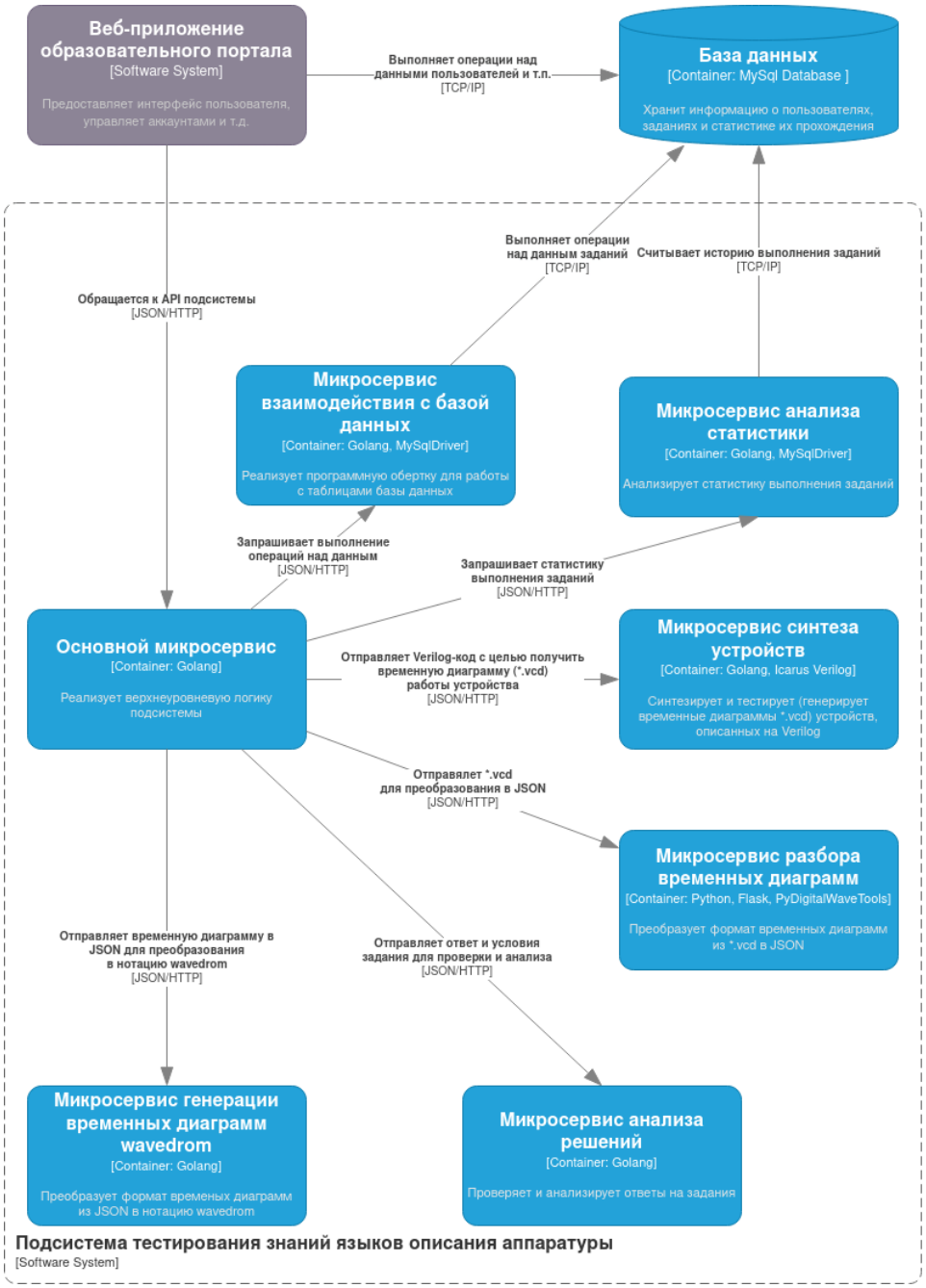

Рис. 3. Контейнер-диаграмма разработанной подсистемы

Проектирование базы данных. После проектирования архитектуры подсистемы была спроектирована структура ее базы данных. В результате анализа предметной области удалось выделить описанные ниже сущности [5].

Сущность «Задание» (таблицы Types, LevelsBrief, LevelsData) — содержит информацию о порядковом номере задания, его условиях, правильном ответе, цене в баллах и т. п.

Сущность «Пользователь» (таблица Users) — позволяет идентифицировать пользователя по ID, определить, обладает ли пользователь правами администратора, и узнать его псевдоним (так называемый никнейм). Кроме того, эта сущность может нести в себе дополнительную информацию, необходимую веб-приложению образовательного портала.

Сущность «Попытка решения» (таблица SolutionEfforts) — содержит информацию об успешности и времени каждой попытки решения задания каким-либо пользователем.

Полученная даталогическая схема базы данных в нотации Мартина изображена на рис. 4.

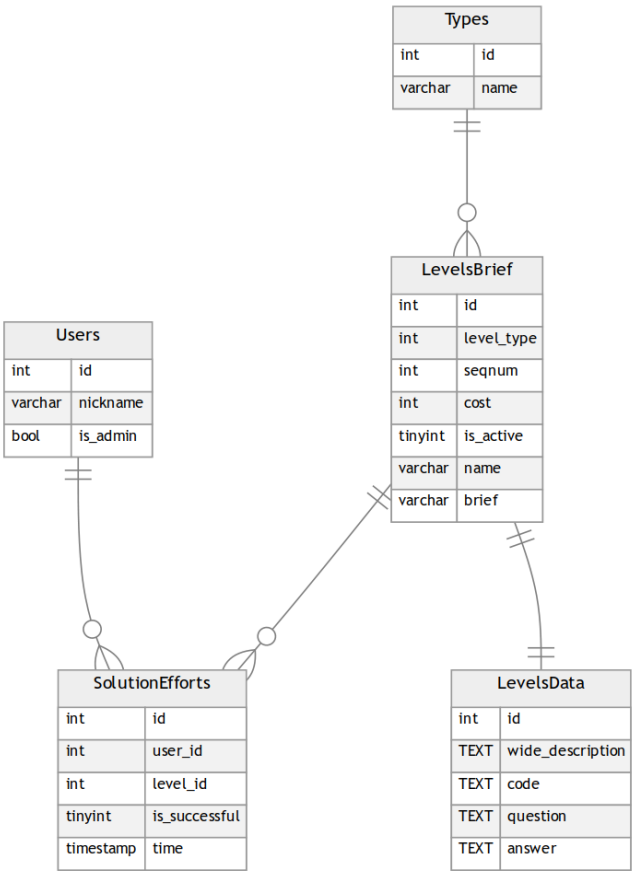

Рис. 4. Даталогическая схема базы данных

Разработка микросервиса синтеза устройств. Поскольку объем статьи не позволяет подробно рассмотреть процесс проектирования всех микросервисов, входящих в подсистему, ниже будет описан только микросервис синтеза устройств и микросервисы для работы с временными диаграммами.

Микросервис синтеза устройств по своей сути является оберткой вокруг программы Icarus Verilog, позволяющей моделировать работу цифровых устройств, описанных на языке Verilog. Программная обертка позволяет передавать в Icarus Verilog по сети исходный код, отправленный обучающимся в качестве ответа на задание в образовательном портале. Достоинствами Icarus Verilog являются [6]:

– малый размер исполняемого файла;

– наличие консольного режима работы (удобно вызывать из программного кода через библиотеки для работы с операционной системой);

– распространение по свободной лицензии (GNU GPL).

Обработка ответа обучающегося осуществляется в несколько этапов, за каждый из которых отвечает свой программный компонент:

– получение HTTP-запроса на симуляцию устройства (класс Router);

– сохранение полученных исходных кодов устрйоства и теста в файловой системе (библиотека OsLib);

– получение временной диаграммы работы устройства (в формате *.vcd) с помощью IcarusVerilog и библиотеки OsExecLib;

– удаление временных файлов;

– отправка HTTP-ответа, содержащего код временной диаграммы.

Диаграмма компоновки микросервиса показана на рис. 5.

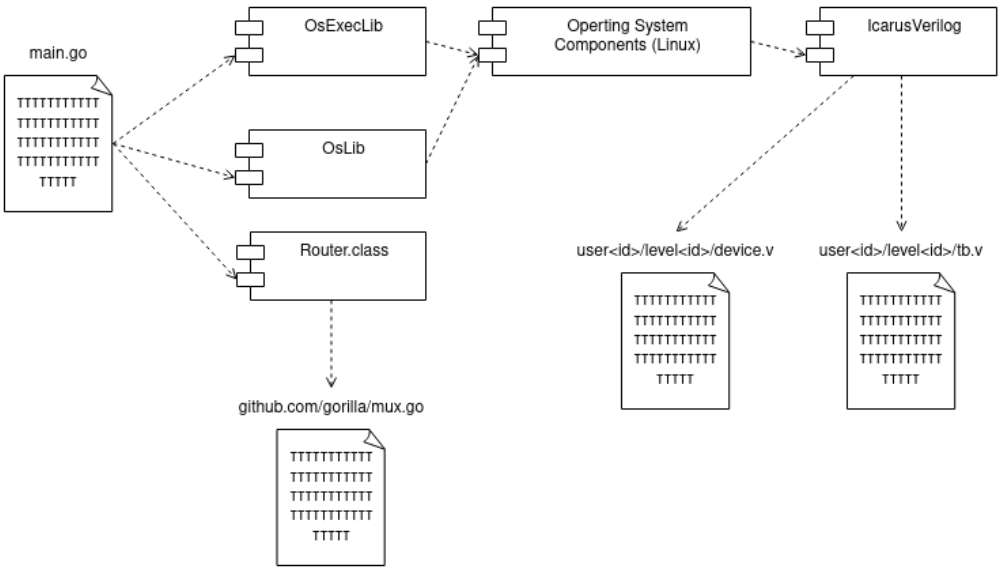

Рис. 5. Диаграмма компоновки микросервиса синтеза устройств

Микросервисы для работы с временными диаграммами. Для преобразования временных диаграмм в подсистеме предусмотрено два микросервиса: микросервис разбора временных диаграмм и микросервис генерации диаграмм wavedrom.

Изначально микросервис синтеза устройств в ходе тестирования работы устройства формирует временную диаграмму в формате *.vcd. Данный формат крайне неудобен как для анализа по сравнению с эталонной временной диаграммой, так и для генерации графического представления временно́й диаграммы в рамках веб-приложения.

Для преобразования временных диаграмм к более удобному для дальнейшей обработки формату был реализован микросервис разбора временных диаграмм. Его исходный код написан на Python с применением библиотеки PyDigitalWaveTools [7]. Данная библиотека преобразует временную диаграмму в формате *.vcd в формат PyDigitalWaveTools согласно алгоритму, заложенному разработчиками библиотеки. Диаграмма Джексона, описывающая этот формат, представлена на рис. 6.

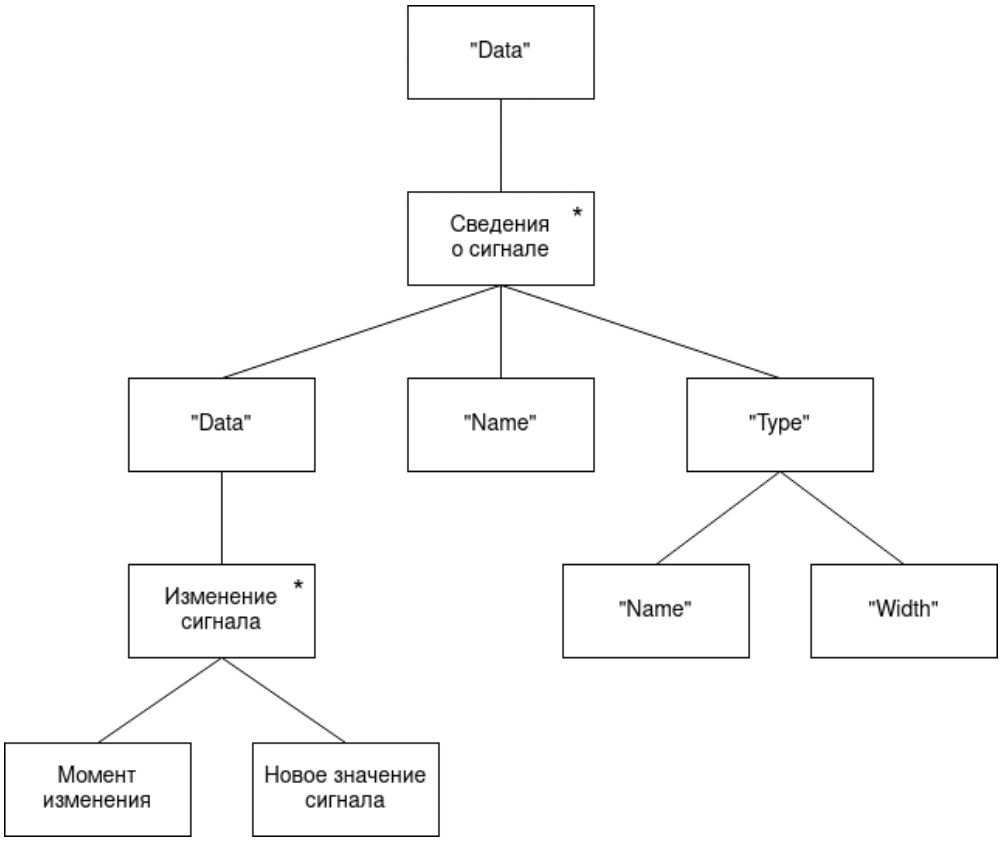

Рис. 6. Формат временных диаграмм в PyDigitalWaveTools

Формат PyDigitalWaveTools намного удобнее для сравнения с эталонной временной диаграммой (в том же формате) и анализа несоответствий, однако алгоритм визуализации для этого формата пришлось бы реализовать самостоятельно. Вместо этого было решено реализовать микросервис генерации временных диаграмм wavedrom, который преобразовал бы временные диаграммы из формата PyDigitalWaveTools в формат движка Wavedrom [8]. Данный движок позволяет визуализировать временные диаграммы посредством http-запроса, содержащего описание сигнала, к специальному интернет-сервису.

Описание формата для движка Wavedrom в нотации Джексона приведено на рис. 7.

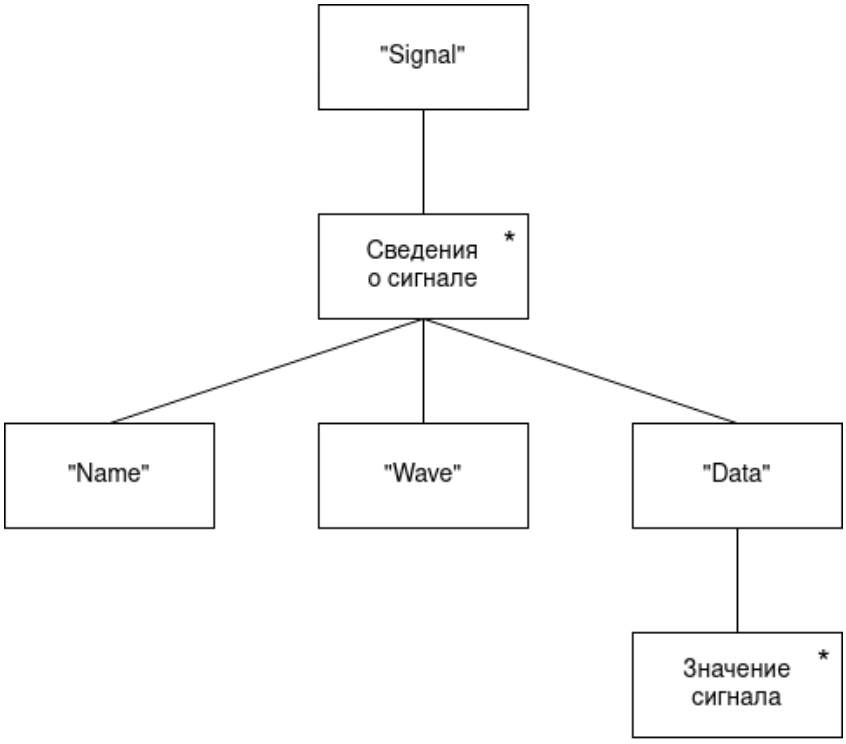

Рис. 7. Формат временных диаграмм для движка Wavedrom

Поля структуры имеют значение, описанное ниже:

– signal — массив всех сигналов временной диаграммы;

– name — имя сигнала;

– wave — форма сигнала (для каждого такта может иметь значения 0, 1, x, z, . — сохранить предыдущее, | — разрыв, = — обратиться к очередному элементу data);

– data — массив, содержащий строковые значения сигнала (можно, например, отобразить большое число для многоразрядной шины).

Суть алгоритма преобразования из формата PyDigitalWaveTools в формат движка Wavedrom состоит в том, чтобы найти наибольший общий делитель для моментов изменения сигналов $t_{НОД}$ и затем провести «дискретизацию» сигналов по времени, просматривая, как изменялся каждый сигнал в моменты времени, кратные $t_{НОД}$.

Тестирование. После проектирования и реализации подсистемы для тестирования знаний языков описания аппаратуры было проведено ее функциональное и нагрузочное тестирование.

Функциональное тестирование проводили по принципу «черного ящика», для автоматизации тестирования применяли библиотеку Pytest [9]. Для создания информативных отчетов о прохождении тестов была использована библиотека Allure [10]. Все 46 функциональных тестов прошли успешно, отчет о прохождении тестов представлен на рис. 8.

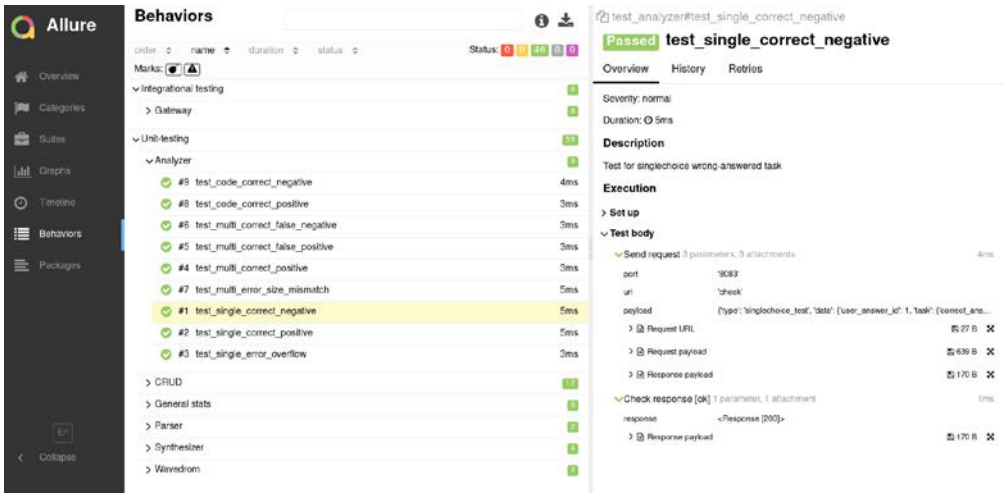

Рис. 8. Отчет о результатах функционального тестирования

Кроме того, было проведено нагрузочное тестирование основного микросервиса, микросервиса синтеза устройств и микросервиса разбора временных диаграмм. Для нагрузочного тестирования использовали библиотеку Locust [11]. От основного микросервиса требовалась средняя задержка отклика не более 500 мс, а от других микросервисов — не более 1000 мс при нагрузке в 100 пользователей. Результаты нагрузочного тестирования основного микросервиса отражены в табл. 3 и 4.

Показатели, полученные в ходе нагрузочного тестирования, находятся в допустимых пределах, следовательно, тестирование прошло успешно.

Таблица 3

Статистика запросов к основному микросервису

Маршрут	Запросы	Число ошибок	Среднее время ответа, мс	Минимальное время ответа, мс	Максимальное время ответа, мс	Средний размер ответа, байт	Запросы/с	Ошибки/с
/levels	6 743	0	85	3	676	506	111,8	0
/stats	20 486	0	161	2	1 151	331	339,5	0
Итого	27 229	0	142	2	1 151	374	451,3	0

Таблица 4

Статистика ответов основного микросервиса

Маршрут	Распределение ответов, мс, по перцентилям (percentile)							
	50	60	70	80	90	95	99	100
/levels	78	94	110	130	160	180	220	680
/stats	110	140	190	270	370	460	630	1 200
Итого	100	130	160	210	330	430	600	1 200

Заключение. В рамках представленной статьи был рассмотрен процесс проектирования и тестирования программной подсистемы для тестирования знаний языков описания аппаратуры. Были рассмотрены варианты использования подсистемы, ее архитектура, симуляция работы цифровых устройств и преобразование временных диаграмм в ее рамках. Также было проведено функциональное и нагрузочное тестирование подсистемы, все тесты завершились успешно.

Литература

[1] Ильина Е.А., Егорова Л.Г., Дьяконов А.В. Технология тестирования знаний студентов с использованием системы. *Математическое и программное обеспечение систем в промышленной и социальной сферах*, 2011, № 1–3, с. 166–172.

[2] Иванова Г.С. *Технология программирования*. Москва, Кнорус, 2016, 333 с.

[3] *The C4 model for visualising software architecture.* URL: https://c4model.com/ (дата обращения 01.02.2023).

[4] Мычко С.И. Микросервисная архитектура. *Информационные технологии. Межвуз. сб. науч. тр*. Рязань, РГРТУ, 2019, с. 166–168.

[5] *Основы правил проектирования базы данных.* URL: https://habr.com/ru/articles/514364/ (дата обращения 05.03.2023).

[6] *Симуляция проекта с помощью Icarus-Verilog.* URL: https://marsohod.org/11-blog/113-icarus (дата обращения 13.03.2023).

[7] *PyDigitalWaveTools.* URL: https://github.com/Nic30/pyDigitalWaveTools (дата обращения 27.03.2023).

[8] *Hitchhiker's Guide to the WaveDrom*. URL: https://wavedrom.com/tutorial.html (дата обращения 03.04.2023).

[9] Зиганшина М.Р., Валиуллина Д.И., Зиганшин И.А. Применение фреймворка Pytest для тестирования программного кода на языке Python. *Ученый XXI века. Междунар. науч.-исследовательский конкурс: сб. ст.* Пенза, Наука и Просвещение, 2022, с. 35–37.

[10] *Allure — фреймворк от Яндекса для создания отчетов автотестов*. URL: https://habr.com/ru/companies/yandex/articles/232697/ (дата обращения 12.04.2023).

[11] *Locust — A modern load testing framework*. URL: https://locust.io/ (дата обращения 18.04.2023).

Астахов Сергей Викторович — бакалавр кафедры «Компьютерные системы и сети», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Лапшин Никита Валерьевич — бакалавр кафедры «Компьютерные системы и сети», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Ким Тамара Александровна, ассистент кафедры «Компьютерные системы и сети», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация. E-mail: kimta@bmstu.ru

Ссылку на эту статью просим оформлять следующим образом:
Астахов С.В., Лапшин Н.В. Программная подсистема тестирования знаний языков описания аппаратуры. *Политехнический молодежный журнал*, 2023, № 06 (83).
http://dx.doi.org/10.18698/2541-8009-2023-6-903

SOFTWARE SUBSYSTEM FOR TESTING KNOWLEDGE OF THE HARDWARE DESCRIPTION LANGUAGES

S.V. Astakhov fzastahov@gmail.com
N.V. Lapshin nikita.lapshin2000@gmail.com

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

The article is devoted to development of a software subsystem for testing knowledge of the hardware description languages, which enables managing the educational materials and automatically checks the tasks, including those on describing the hardware devices in the Verilog language. The existing knowledge testing systems were analyzed. In the course of this analysis, functional requirements were formulated, and a diagram of options for using the software subsystem for testing knowledge of the hardware description languages was compiled. The subsystem architecture and components were designed. Microservice architecture was used in development, most of the microservices were implemented in the Golang programming language. In addition, the Python language and the PyDigitalWaveTools library were used to work with the timing diagrams. The Icarus Verilog program was used to simulate the digital devices behaviour. The developed subsystem was exposed to functional and load testing.

Keywords

Knowledge testing, hardware description language, HDL, Verilog, distance learning system, educational portal, microservice architecture, Golang, functional testing, load testing

Received 01.06.2023

References

[1] Il'ina E.A., Egorova L.G., D'yakonov A.V. Moodle Technology testing knowledge of students using system moodle. *Matematicheskoe i programmnoe obespechenie sistem v promyshlennoy i sotsial'noy sferakh*, 2011, no. 1–3, pp. 166–172. (In Russ.).

[2] Ivanova G.S. *Tekhnologiya programmirovaniya* [Programming technology]. Moscow, Knorus Publ., 2016, 333 p. (In Russ.).

[3] *The C4 model for visualising software architecture*. URL: https://c4model.com/ (accessed February 1, 2023).

[4] Mychko S.I. Microservice architecture. *Informatsionnye tekhnologii. Mezhvuz. sb. nauch. tr.* [Information Technology. Interuniversity collection of scientific papers]. Ryazan, RSREU Publ., 2019, pp. 166–168. (In Russ.).

[5] *Osnovy pravil proektirovaniya bazy dannykh* [Fundamentals of database design rules]. URL: https://habr.com/ru/articles/514364/ (accessed March 5, 2023).

[6] *Simulyatsiya proekta s pomoshch'yu Icarus-Verilog* [Project simulation with Icarus-Verilog]. URL: https://marsohod.org/11-blog/113-icarus (accessed March 13, 2023).

[7] *PyDigitalWaveTools*. URL: https://github.com/Nic30/pyDigitalWaveTools (accessed March 27, 2023).

[8] *Hitchhiker's Guide to the WaveDrom*. URL: https://wavedrom.com/tutorial.html (accessed April 3, 2023).
[9] Ziganshina M.R., Valiullina D.I., Ziganshin I.A. Python Application of the Pytorch library for training neural networks. *Uchenyy XXI veka. Mezhdunar. nauch.-issledovatel'skiy konkurs: sb. st*. [Scientist of the 21st century. International research competition: collection of articles]. Penza, Nauka i Prosveshchenie Publ., 2022, pp. 35–37. (In Russ.).
[10] *Allure — freymvork ot Yandeksa dlya sozdaniya otchetov avtotestov* [Allure is a framework from Yandex for creating autotest reports].
URL: https://habr.com/ru/companies/yandex/articles/232697/ (accessed April 12, 2023).
[11] *Locust — A modern load testing framework*. URL: https://locust.io/ (accessed April 18, 2023).

Astakhov S.V. — Bachelor's Program Student, Department of Computer Systems and Networks, Bauman Moscow State Technical University, Moscow, Russian Federation.

Lapshin N.V. — Bachelor's Program Student, Department of Computer Systems and Networks, Bauman Moscow State Technical University, Moscow, Russian Federation.

Scientific advisor — Kim T.A., Assistant, Department of Computer Systems and Networks, Bauman Moscow State Technical University, Moscow, Russian Federation. E-mail: kimta@bmstu.ru

Please cite this article in English as:
Astakhov S.V., Lapshin N.V. Software subsystem for testing knowledge of the hardware description languages. *Politekhnicheskiy molodezhnyy zhurnal*, 2023, no. 06 (83). (In Russ.). http://dx.doi.org/10.18698/2541-8009-2023-6-903